UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

HENRIQUE RICARDO FIGUEIRA

FIREWALL PARA REDES LORAWAN

CAMPO MOURÃO

2022

HENRIQUE RICARDO FIGUEIRA

FIREWALL PARA REDES LORAWAN

Firewall for LoRaWAN Networks

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Luiz Arthur Feitosa Dos Santos

Coorientador: Prof. Ms. Paulo Cesar Gonçalves

CAMPO MOURÃO

2022

HENRIQUE RICARDO FIGUEIRA

FIREWALL PARA REDES LORAWAN

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 27/outubro/2021

Prof. Dr. Rodrigo Campiolo
Doutorado
UTFPR

Prof. Dr. Rafael Liberato Roberto
Doutorado
UTFPR

Prof. Ms. Paulo Cesar Gonçalves
Mestrado
UTFPR

CAMPO MOURÃO

2022

Para Vitória que nunca desistiu de mim, para
meu pai e para minha mãe...

AGRADECIMENTOS

Este trabalho é resultado de muito esforço e dedicação, agradeço a todos que contribuíram para o desenvolvimento deste trabalho. Gostaria de agradecer:

A minha família, em especial a minha namorada Vitória Rorato Rufino, meus pais Hamilton Gutierrez Figueira e Maysa Ricardo da Silva Figueira, que não mediram esforços para que eu realizasse este trabalho e nos momentos mais difíceis estiveram do meu lado.

Ao meu orientador Luiz Arthur Feitosa dos Santos que ao longo deste processo me auxiliou, aconselhou, sempre esteve disposto a me ajudar com muita paciência e respeito.

Aos membros da banca Paulo Cesar Gonçalves, Rodrigo Campiolo e Rafael Liberato que prontamente aceitaram o convite e ajudaram a contribuir para este trabalho.

Aos meus amigos João Gris, Giovani Bertuzzo, Gabriel Negrão, Higor Celante, João Gimenez que estiveram comigo desde o inicio do curso. Também ao Victor Benso que auxiliou neste trabalho e é uma inspiração profissional.

A todos os funcionários da UTFPR-CM que sempre me trataram muito bem, em especial ao porteiro Fabiano Coutinho pelas conversas e risadas nos intervalos das aulas. Também gostaria de agradecer ao Devair Pereira(Tio Do Pastel) que sempre fortaceleu com um café.

RESUMO

Este trabalho aborda questões de segurança em protocolos IoT com a intenção de prevenir e mitigar ataques cibernéticos em redes de rádio frequência que permitem comunicação de longa distância com baixo consumo de energia, conhecidas como redes LoRa. O principal objetivo foi o desenvolvimento e avaliação de um *firewall* que bloqueia pacotes LoRa, também foi analisado se solução proposta prejudicaria o funcionamento normal do *gateway*. Para o desenvolvimento do *firewall* foi necessário analisar como o *software RisingHF-gateway* realizava o encaminhamento de pacotes e realizar alterações no código. Após o desenvolvimento do *firewall* foi possível constatar que é factível bloquear pacotes LoRa através do endereço de dispositivos e também que a solução implementada não interfere no funcionamento normal do código. O resultado da pesquisa foi o desenvolvimento de um *firewall* com regras que permitem bloquear ou autorizar a retransmissão de pacotes de um determinado dispositivo.

Palavras-chave: iot; lora.

ABSTRACT

This work addresses security issues in IoT protocols to prevent and mitigate cyberattacks on radio frequency networks that allow long-distance communication with low energy consumption, known as LoRa networks. The main objective was the development and evaluation of a *firewall* that blocks LoRa packets; it was also analyzed whether the proposed solution would harm the normal functioning of the gateway. To develop the *firewall* was necessary to analyze how the RisingHF-gateway software carried out the forwarding of packets and make changes to the code. After the development of the *firewall*, it was possible to verify that it is feasible to block LoRa packets through the device address and that the implemented solution does not interfere with the normal functioning of the code. The result of the research was the development of a *firewall* with rules that allow blocking or authorizing the retransmission of packets from one device.

Keywords: iot; lora.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

ABP	*Activation by Personalization*
AES	*Advanced Encryption Standard*
ANATEL	Agência Nacional de Telecomunicações
CPU	*Central Processing Unit*
DDoS	*Distributed Denial of Service*
DoS	*Denial of Service*
FHDR	*Frame Header*
GHz	Giga-hertz
IoT	*Internet of Things*
IP	*Internet Protocol*
ISM	*Industrial Scientific and Medical*
JSON	*JavaScript Object Notation*
Kbps	Kilobits por segundo
KM	Quilômetros
LoRa	*Long-Range*
LoRaWAN	*LoRa for Wide Area Network*
LPWAN	*Low Power Wide Area Network*
M2M	*Machine to Machine*
MAC	*Media Access Control*
MHDR	*MAC Header*
MHz	Mega-hertz
NAT	Network Address Translation

OTAA	*Over-the-Air Activation*
PHY	*Physical Layer*
RF	*Radio frequency*
RFID	*Radio-Frequency Identification*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
UTFPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

O termo *Internet of Things* (IoT) vem chamando a atenção da comunidade nos últimos tempos, trazendo o conceito de conectar “coisas” na Internet para transmitir dados, em que essas coisas podem ser desde câmeras de segurança ou até cafeteiras “inteligentes”. Em 2020 havia aproximadamente 10 bilhões de dispositivos IoT conectados à Internet, estima-se que haverá 30 bilhões em 2025 (MALAN *et al.*, 2020). Levando em consideração estas informações, uma questão que pode ser levantada é: como interconectar todos esses aparelhos?

O protocolo *LoRa for Wide Area Network* (LoRaWAN) pode ser a resposta desta questão, baseado na tecnologia LoRa, rede de radiofrequência da família de redes de longas distância e baixa frequência. Possui baixo consumo energético e pode transmitir para grandes distâncias, alcançando 5 Quilômetros (KM) nas áreas urbanas e até 15 km em áreas rurais (Centenaro *et al.*, 2016). O protocolo LoRaWAN, que surgiu em 2015, vem ganhando cada vez mais mercado ao redor do mundo (LORA ALLIANCE, 2021c).

O sucesso de redes IoT e o acesso a uma quantidade enorme de informações vem atraindo a atenção de pessoas mal intencionadas. Em 2020, os ataques a dispositivos IoT aumentaram 66% com relação ao ano anterior e totalizaram mais de 50 milhões ataques, somente em outubro de 2020 foram mais de 10 milhões de ataques (Atlas VPN, 2021b). O resultado disso foi que os ataques cibernéticos custaram ao mundo mais de 1 trilhão de dólares em 2020, o que representa cerca de 1% do produto interno bruto global (Atlas VPN, 2021a).

Tendo em vista que as tecnologias utilizadas para o crescimento das redes IoT são relativamente novas, muitas vulnerabilidades ainda não foram corrigidas, ou mesmo identificadas, trazendo preocupação com a segurança dos dispositivos (Yang *et al.*, 2018).

O funcionamento do protocolo LoRaWAN permite que dispositivos transmitam para um ou mais *gateways* em seu alcance, assim melhorando a transmissão (LORA ALLIANCE, 2021b). Haja vista esta afirmação conclui-se que os *gateways* podem receber pacotes de qualquer dispositivo que esteja em seu alcance, sendo assim pessoas maliciosas podem usar esta informação para ordenar ataques a dispositivos finais ou a *gateways*. Segundo Yang *et al.* (2018), ataques como *Denial of Service* (DoS) e *ACK Spoofing* podem ser realizados em dispositivos LoRaWAN devido a vulnerabilidades presentes no protocolo. Assim se faz necessário que haja algum mecanismo que atue no *gateway* para autorizar ou bloquear a retransmissão de pacotes visando prevenir e mitigar ataques cibernéticos a redes LoRaWAN.

Desta forma o presente trabalho tem como objetivo desenvolver um *firewall* para redes LoRaWAN com a finalidade de bloquear/liberar pacotes e proporcionar maior segurança a essas redes. Pois até o momento do desenvolvimento deste trabalho não há nenhum *firewall* para este tipo de rede.

Após o desenvolvimento do *firewall* foram realizados experimentos, em cenários de testes com dispositivos reais, para verificar se a implementação do *firewall* aumentava o tempo de

retransmissão do pacote. Os resultados mostram que o funcionamento da rede não é afetado pelo *firewall* .

2 CONCEITOS

Este capítulo aborda os conceitos e definições a respeito da base teórica e tecnologias utilizadas no desenvolvimento do *firewall* para redes LoRaWAN. A Seção 2.1 conceitua a respeito de cibersegurança e *firewalls*, já a Seção 2.2 contextualiza IoT, suas características e tecnologias que são utilizadas para viabilizar a implementação da mesma, como: *Low Power Wide Area Network* (LPWAN), LoRa, LoRaWAN.

2.1 CiberSegurança

As redes de computadores, que inicialmente não possuíam muitos usuários e geralmente eram usadas para realizar pesquisa universitária e envio de correio eletrônico. Com o avanço da tecnologia, algumas tarefas do nosso cotidiano foram automatizadas e a utilização da Internet tornou-se algo indispensável no dia a dia. Tendo em vista todos esses fatores e como forma de deter a expansão do nível de ameaça, surgiu uma crescente preocupação em proteger computadores, servidores, dispositivos móveis, sistemas eletrônicos, redes e dados de ataques (TANENBAUM; WETHERALL, 2011).

A segurança da informação baseia-se em princípios para proteção contra ameaças, esses conceitos são confidencialidade, integridade e disponibilidade (TANENBAUM; WETHERALL, 2011). A confidencialidade limita o acesso de informação a pessoas previamente autorizadas. A integridade garante que as informações não sofreram interferência externa que possam corromper as mesmas. Por fim, a disponibilidade exige que as informações necessitam estar disponíveis, a qualquer momento (TANENBAUM; WETHERALL, 2011).

Uma parte pertencente a segurança da informação é a cibersegurança que traz como característica o monitoramento continuo de recursos eletrônicos a fim de combater o acesso não autorizado a serviços remotos, identificar a autenticidade de mensagens que trafegam nas redes e impedir que pessoas não autorizadas acessem dados que não as pertencem.

As ameaças combatidas pela cibersegurança são: crimes virtuais, que são atividades ilegais realizadas com o auxilio da tecnologia; Há também a guerra cibernética, na qual a principal motivação é a aquisição de dados para fins políticos; e por último o terror virtual no qual se instala o pânico e medo nos usuários comuns (KASPERSKY, 2021a).

Uma dificuldade que a cibersegurança encontra, é a variedade de técnicas de ataque contra a segurança dos dados, um tipo de ataque muito utilizado nos dias de hoje é o *phishing* que como Avast (2021) define, é qualquer tipo de fraude utilizando meios de telecomunicação, empregando engenharia social para obter dados das vitimas. Já o DoS e *Distributed Denial of Service* (DDoS) trabalham de outra forma, enviando vários pacotes de rede para o mesmo local, sobrecarregando *hosts* e com isso impedindo o seu funcionamento normal (RASH, 2007). Outra técnica bem conhecida nesse meio é o ataque de força bruta que aplica senhas aleatoriamente até identificar a sequência correta para ter acesso a dados não autorizados. Para que esses

ataques sejam mitigados muitas empresas utilizam de uma técnica conhecida de segurança, chamado *Firewall* (COMER, 2016).

A cibersegurança é um campo muito vasto, portanto há outros problemas de segurança e também diversos métodos de defesa. Para contexto deste trabalho, iremos abordar somente *firewalls*.

2.1.1 Firewall

O termo em inglês *firewall* compara a técnica de segurança com uma parede antechamas que evita que o fogo e a fumaça se alastre para outros cômodos em construções. Basicamente sua função, é atuar entre a redes, tais como: rede local e Internet, extranet e intranet, etc. Inspecionando o tráfego na rede e decidindo o que entra e o que é descartado, de acordo com um conjunto de políticas e regras de segurança determinadas pelo administrador da rede. O *firewall* pode ser implementado tanto via *hardware*, quanto *software* ou ambos (KASPERSKY, 2021b).

O *firewall* não é um programa e sim um conjunto de recursos destinados a manter a rede segura. Segundo Moraes (2015), as principais funções dos *firewalls* são:

- Estabelecer um perímetro de segurança;
- Separar as redes e controlar acessos;
- Aumentar a privacidade;
- Registrar e gerar estatísticas do uso da rede e acessos indevidos;
- Proteger sistemas vulneráveis na rede.

Existem alguns tipos de *firewalls*, os filtros de pacotes que verificam os endereços de *Internet Protocol* (IP) e as portas *Transmission Control Protocol* (TCP)/*User Datagram Protocol* (UDP) de acordo com uma lista de controle, permitindo ou bloqueando a entrada na rede. De acordo com Moraes (2015), são algumas vantagens desse tipo de *firewall*: rapidez, eficiência, transparência e flexibilidade. O que pode ser considerado uma desvantagem é que esta implementação requer muitos testes para verificação das funcionalidades. Guerra (2019), descreve uma ferramenta para auxiliar este tipo de configuração e tornar a implementação de regras simples e prática. Outro tipo de *firewall* é com inspeção de estado que monitora conexões desde da abertura até o fechamento e libera ou bloqueia pacotes de rede de acordo com seu estado, porta ou protocolo. Suas decisões são tomadas de acordo com regras implementadas por administradores e por históricos de conexões anteriores. Também existe o *firewall* de gerenciamento unificado de ameaças, que consiste em uma única solução de segurança mesclando funções do *firewall* de inspeção de estado com outras funções como prevenção de intrusões, antivírus e roteamento remoto (CISCO, 2021b).

A seção a seguir aborda alguns conceitos sobre IpTables, pois no presente trabalho será desenvolvido um *firewall* de filtro de pacotes com a finalidade de restringir ou liberar pacotes que trafegam por *gateways* LoRaWAN.

2.1.2 IpTables

Desenvolvido pelo projeto NetFilter (NETFILTER, 2021), o IpTables é um *firewall* de filtragem de pacotes presente no núcleo do Linux desde janeiro de 2001, que utiliza a estrutura NetFilter para executar operações em pacotes. Ele permite a criação de políticas que fornecem ao usuário controle sobre os pacotes IP que comunicam-se a sistemas Linux. O IpTables atua bloqueando pacotes que não possuem permissão para trafegar e libera pacotes que se enquadram nas regras vigentes (RASH, 2007).

As políticas pertencentes ao IpTables são construídas a partir de um conjunto ordenado de regras, que descrevem para o núcleo ações que devem ser tomadas (RASH, 2007). Essas regras são aplicadas em tabelas, o Iptable possui quatro tabelas:

- *Filter* - Tabela padrão que determina liberação/bloqueio de pacotes de rede.
- Network Address Translation (NAT) - Usada para gerenciar a tradução dos endereços que trafegam na rede.
- *Mangle* - Utilizada para aplicar regras especificas que alteram dados de pacotes, como modificações no cabeçalho de um pacote (RASH, 2007).
- *Raw* - Utilizada principalmente para definir pacotes que não devem ser manipulados pelo sistema de rastreamento de conexões.

Cada tabela possui um conjunto de *chains*, conhecidas como situações, que são utilizadas para construir regras. A tabela *Filter* possui as seguintes *chains*:

- *INPUT* - Aplicada a pacotes da rede que são destinados ao *host*.
- *OUTPUT* - Reservada a pacotes de rede locais.
- *FORWARD* - Gerencia pacotes que são roteados pelo *host*.

Já as *chains* que a tabela NAT possui são:

- *PREROUTING* - Altera pacotes quando são recebidos.
- *OUTPUT* -Altera pacotes locais antes de serem enviados.
- *POSTROUTING* - Altera pacotes antes de serem enviados.

A tabela *Mangle* especifica ações especiais no tráfego que passa pelas *chains*, elas ocorrem antes das *chains* das tabelas *Filter* e NAT. As *chains* que pertencem a tabela *Mangle* são a junção das *chais* das outras tabelas e as regras que estão nelas são aplicadas antes de qualquer outra (HAT, 2021).

A seguir, abordam-se conceitos sobre IoT e tecnologias utilizadas para implementar este tipo de rede como LPWAN, LoRa e LoRaWAN.

2.2 IoT - Internet das Coisas

O termo Internet das Coisas foi utilizado pela primeira vez por Kevin Ashton em 1999 na indústria e englobava somente que dispositivos de identificação por radiofrequência *Radio-Frequency Identification* (RFID) (ASHTON, 2009). Porém, com o passar do tempo a definição passou a englobar várias outras áreas como transportes, saúde, ecologia, entre outros. Apesar do termo “coisas” mudar com o avanço da tecnologia, o principal objetivo é interagir com o ambiente sem intervenção humana. Com a evolução do termo IoT inúmeras novas tecnologias foram se incorporando a este tipo de rede, tais como Wi-Fi, Bluetooth, comunicações de máquina para máquina *Machine to Machine* (M2M), entre outras (Palattella *et al.*, 2016). Ainda hoje, IoT não apresenta uma definição padrão, por exemplo, para Gubbi *et al.* (2013) a definição de IoT que não diz respeito a nenhum protocolo de comunicação e é apresentada a seguir:

Interconexão de dispositivos de detecção e atuação, oferecendo a capacidade de compartilhar informações entre plataformas por meio de uma estrutura unificada, desenvolvendo uma imagem operacional comum para permitir aplicativos inovadores. Isso é conseguido através da detecção onipresente e contínua, análise de dados e representação de informações com a computação em nuvem como a estrutura unificadora (GUBBI *et al.*, 2013).

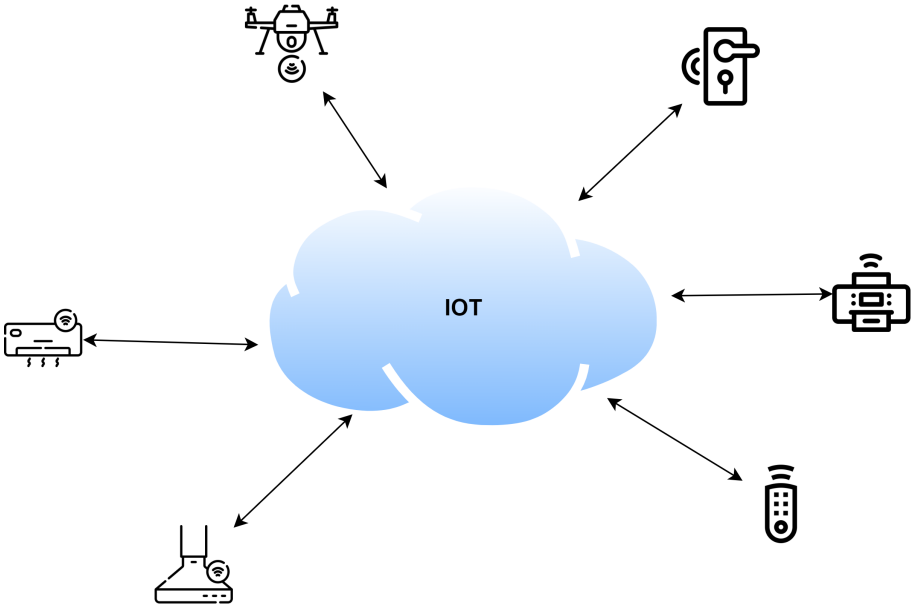
IoT são redes de objetos interconectados que não apenas coletam informações do ambiente mas também interagem com o mundo físico e utiliza a Internet para fornecer serviços de análises e transferência de informações (Palattella *et al.*, 2016).

Ao longo dos anos, os procolos que apresentam características de transmissão de curto alcance como ZigBee (SIGFOX, 2021) e Bluetooth se tornaram as aplicações mais viáveis para implantação de serviços IoT. Porém, essa característica dificulta o desenvolvimento de serviços que exigem transmissões de longa distância como cidades inteligentes (Palattella *et al.*, 2016). Tendo em vista esse problema, outra proposta foi a comunicação via redes de celulares, entretanto também não obteve-se sucesso, pois elas não foram projetadas para receber um grande número de dispositivos M2M, como sensores e atuadores. A comunicação IoT necessita da transmissão esporádica de pequenos pacotes e vários dispositivos M2M interligados ao mesmo tempo, para isso uma alternativa tornou-se viável, a utilização de redes de longo alcance e baixo consumo, tal como LPWAN (Meddeb, 2016).

Com o avanço da tecnologia, a proposta de redes IoT vem ganhando cada vez mais força, segundo (CISCO, 2021a) em 2023 haverá 14.7 bilhões de conexões máquina para má-

quina que representa 1.8 conexões para cada humano na terra, essas conexões podem estar presentes em impressoras, fechaduras, sensores, exaustores, *drones*, entre outros como a Figura 1 ilustra. Esses dispositivos possuem *hardware* limitado o que não permite a instalação de nenhum ant-vírus. Levando em consideração que em 2020 a América Latina sofreu mais de 41 bilhões de tentativas de ataques cibernéticos (Fortinet, 2021) é necessário refletir a respeito da segurança deste tipo de rede, pois nada impede de dispositivos maliciosos conectarem a rede e tentarem algum ataque. Sabendo disso se faz necessário algum tipo de proteção a parte do protocolo, uma solução seria a implantação de *firewalls* para uma proteção eficaz. Até o momento deste trabalho não existe implementações de *firewalls* para redes LoRa.

Figura 1 – Exemplo de rede IoT.



Fonte: Autoria própria (2022).

As seções seguintes explicam conceitos a respeito de redes LPWAN e tecnologias que surgiram a partir dela, como LoRa e LoRaWAN.

2.2.1 LPWAN - *Low Power Wide Area*

As redes LPWAN preenchem a lacuna das tecnologias tradicionais de comunicação sem fio de curto alcance e atendem aos diferentes requisitos de aplicações IoT, com transmissões intermitentes e esporádicas de pequenos pacotes de dados, tolerantes a atrasos e perda de pacotes (Centenaro *et al.*, 2016).

O principal intuito das redes LPWAN é fornecer comunicações de longo alcance conectando dispositivos distribuídos por uma grande área geográfica. Suas principais características são garantir uma grande área de cobertura, na ordem de dezenas de quilômetros, um baixo custo de manutenção, baixas taxas de transmissão com pequenos pacotes de dados e operar em espectros de frequência *Industrial Scientific and Medical* (ISM), 169 Mega-hertz (MHz), 433

MHz, 868/915 MHz e 2,4 Giga-hertz (GHz), dependendo da região de operação. No Brasil as faixas de frequência definidas pela Agência Nacional de Telecomunicações (ANATEL) são: 902-907,5 MHz e 915-928 MHz (Lavric; Petrariu, 2018). Uma característica importante é que os nós são conectados diretamente a uma ou mais estações base, muito semelhante as rede de celular clássicas, simplificando assim a cobertura de grandes áreas, pois a forma como foi projetada sua camada física proporciona uma sensibilidade muito alta ao receptor. Além disso, a topologia das LPWAN possibilita maior controle de latência da conexão, visto que os dispositivos estão ligados diretamente ao *gateway*. Sendo mais adequada para suportar requisitos das cidades inteligentes (Centenaro *et al.*, 2016).

Um dos percursores do desenvolvimento de redes LPWAN foi a Sigfox (Connectivity Standards Alliance, 2021), operadora de rede francesa, que deu início a esta tecnologia em um período em que os dispositivos de rádio estavam em decadência. O sucesso da tecnologia criada movimentou a industria da época e logo foram seguidos por outros grupos como Ingenu, lora Alliance e empresas de telefonia (CHAUDHARI; ZENNARO; BORKAR, 2020). Existem várias soluções LPWAN disponíveis no mercado, uma delas é o LoRa, que é apresentado na seção a seguir.

2.2.2 LoRa

Uma das soluções mais populares para implementação de redes IoT e pertencente a família das LPWANs, que vem ganhando mercado é o LoRa. Uma tecnologia de rádio frequência que permite comunicação a longa distância com baixo consumo de energia. Pode atingir grandes distâncias tanto na área rural quanto na área urbana, esta tecnologia vem se popularizando e conquistando cada vez mais espaço no mercado, pois um alto alcance de transmissão e possui um baixo custo de implementação.

O LoRa é um protocolo, projetado e patenteado pela Semetch Corporation, de modulação por espelhamento espectral que troca taxa de dados por sensibilidade dentro de uma largura de banda de canal fixo, segundo a qual um símbolo é codificado em uma sequência mais longa de bits, reduzindo a interferência e sem alterar a frequência (SEMTECH, 2015). Este esquema visa ser utilizável em dispositivos de bateria de longa duração, no qual o consumo de energia é de suma importância. A camada física LoRa permite comunicações de longo alcance, baixa potência, baixa taxa de transferência (até 50 Kilobits por segundo (Kbps)) e pode operar em 169 MHz, 433 MHz e 915 MHz, nos EUA. Na Europa funciona na faixa de 868 MHz e no Brasil as faixas de frequência são 902-907,5 MHz e 915-928 MHz (AUGUSTIN *et al.*, 2016).

A rede LoRa é apresentada com topologia estrela, no qual os nós são conectados via LoRa *link* a uma ou várias estações base (*gateways*), que estão conectados a servidores por meio do protocolo UDP (Centenaro *et al.*, 2016). Os dispositivos finais não precisam estar associados a um *gateway* específico para obter acesso a rede e sim a um servidor. Os *gateways* não são visíveis aos dispositivos finais, simplesmente encaminham para o servidor associado men-

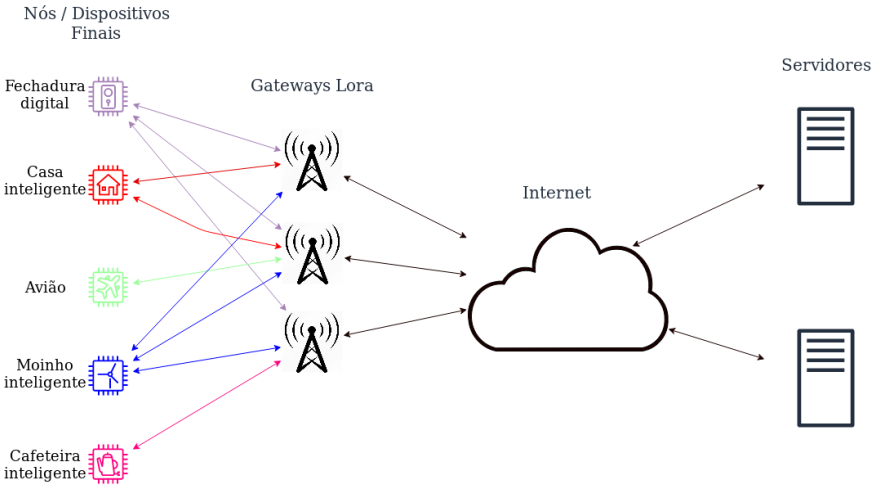
sagens decodificadas enviadas por dispositivos e algumas informações a respeito da qualidade da transmissão (Centenaro *et al.*, 2016). A pilha de protocolos LoRa é dividida em duas camadas, a camada *Physical Layer* (PHY) que é de propriedade da Semtech e tem alguns detalhes ocultos do público em geral, por se tratar de um protocolo privado, e a camada LoRaWAN que é mantida aberta pelos desenvolvedores. Este protocolo é discutido detalhadamente na seção a seguir.

2.2.3 LoRaWAN

LoRaWAN é um protocolo *Media Access Control* (MAC), para redes de longo alcance podendo atingir distâncias de até 10 KM em áreas urbanas e 30 km em áreas rurais, mantido pela lora Alliance (ALLIANCE, 2021), uma associação sem fins lucrativos formada por diversas empresas do ramo como IBM, Actility, Semtech e Microchip.

O protocolo LoRaWAN foi projetado para permitir que dispositivos de baixo consumo de energia, como sensores e/ou atuadores, se comuniquem com aplicativos conectados à Internet, em conexões sem fio de longo alcance. Ele é implementado utilizando topologia estrela (LORA ALLIANCE, 2021a), ou seja, cada nó é conectado a *gateways*, nos quais os *gateways* retransmitem mensagens entre os nós e um servidor de rede central. Os *gateways* são conectados ao servidor de rede por meio de conexões IP e atuam convertendo pacotes *Radio frequency* (RF) em pacotes IP, e vice-versa. A comunicação sem fio aproveita as características de longo alcance da camada física LoRa, permitindo conexões diretas entre o dispositivo final e um ou vários *gateways* (ALLIANCE, 2021). A Figura 2 ilustra da arquitetura LoRaWAN.

Figura 2 – Arquitetura LoRaWAN



Fonte: Autoria própria (2022).

Conforme ilustrado na Figura 2, a arquitetura LoRaWAN possui:

- Dispositivos Finais: São sensores/atuadores conectados a uma rede LoRaWAN por meio de estações base (*gateways*) utilizando modulação RF LoRa. Esses objetos pos-

suem comunicação de baixa potência incorporando conceitos básicos de redes, também conhecidos como nós, podem ser sensores de temperatura, leitores de consumo de energia, alarmes, entre outros. A Figura 3 refere-se ao sensor de temperatura e umidade do modelo Dragino LHT65, que é um exemplo de dispositivo final LoRa. Geralmente os dispositivos finais LoRa são autônomos, operados por bateria e quando fabricados recebem identificadores exclusivos que são utilizados para ativação, administração e segurança (SEMTECH, 2021);

Figura 3 – Sensor de temperatura e umidade LoRaWAN.

Fonte: Seeed Technology (2022).

- *Gateways* LoRaWAN: Dispositivos que recebem mensagens RF de nós que estão em seu alcance e as encaminha para o servidor, por isso servem como ponte entre o servidor e dispositivos finais. Como não há associação fixa entre um dispositivo final e um *gateway*, o mesmo dispositivo final pode se comunicar com todos os *gateway* que estão em seu alcance (SEMTECH, 2021). Essa característica reduz taxas de erros e consequentemente a sobrecarga da bateria para dispositivos finais móveis, também proporciona geolocalização de baixo custo, desde que o *gateway* esteja habilitado para tal (SEMTECH, 2021). Os *gateways* LoRaWAN operam inteiramente na camada física e o tráfego de IP para o servidor pode ser via *Ethernet*, conexão celular ou Wi-Fi. No decorrer deste texto a menção a palavra *gateway* se refere ao *gateway* LoRaWAN utilizado para implementação do trabalho;
- Servidor de Rede: Responsável por receber pacotes dos *gateways* e executa ações específicas, como filtrar pacotes duplicados ou indesejados e responder aos dispositivos finais (Centenaro *et al.*, 2016).

O LoRaWAN utiliza diferentes classes de dispositivos finais que são associadas a modos operacionais diferentes e otimizam uma variedade de perfis de aplicativos. Existem três tipos de classes:

- Classe A: permite transmissões bidirecionais na qual cada transmissão *uplink*, mensagens enviadas pelos nós ao servidor, é seguida de dois curtos períodos recepção de dados do servidor, que são conhecidos como janelas de *downlink*, mensagens enviadas do servidor para um nó, aguardando comandos ou pacotes de dados fornecidos pelo servidor. O nó pode entrar em modo de suspensão com baixo consumo de energia por tempo definido previamente e não há necessidade de rede para reativação. Geralmente destinada para aplicações de monitoramento, em que os dados produzidos pelos nós devem ser coletados por uma estação de controle (Centenaro *et al.*, 2016);

- Classe B: Os dispositivos de classe B possuem uma janela de recepção extra, em relação a classe A, com tempos fixos. A principal ideia desta classe é ter o dispositivo disponível para recepções agendadas. Isso permite que o servidor saiba quando o dispositivo final está escutando, geralmente destinada a nós que precisam receber comandos de um controlador remoto (Centenaro *et al.*, 2016);

- Classe C: Os dispositivos classe C possuem praticamente uma janela de recepção aberta, estando fechada somente quando está transmitindo, indicada para dispositivos diretamente conectados a rede elétrica devido ao seu alto consumo de energia (ALLIANCE, 2021);

Os dispositivos classe C devem oferecer suporte a todas as classes, os classe B também oferecem suporte a A (ALLIANCE, 2021).

Como é possível observar, o LoRaWAN é um protocolo bem definido e fácil de ser aplicado em ambientes reais podendo ser configurado em ambientes que não possuem fácil acesso, como fazendas e áreas remotas. A seção a seguir aborda o tópico de segurança em redes IoT.

2.2.4 Segurança para IoT

Sabendo dos riscos de ataques a redes IoT, uma das maiores preocupações é fornecer segurança estes tipos de implementações. Para fornecer confidencialidade e integridade, o LoRaWAN utiliza um par de chaves criptografadas distintas de 128 bits em conjunto com o algorítimo *Advanced Encryption Standard* (AES), a chave de rede *NwkSKey* e a chave do aplicativo *AppSKey*. Existem dois mecanismos para o procedimento de ativação, *Activation by Personalization* (ABP) e *Over-the-Air Activation* (OTAA) (Yang *et al.*, 2018).

O procedimento ABP possui parâmetros exclusivos *DevAddr*, *NwkSKey*, *AppSKey* que são atribuídos ao dispositivo final e guardados no servidor. Quando há troca de mensagens entre o dispositivo e o servidor, essas mensagens são criptografadas e assinadas e só podem ser lidas pelo servidor. Caso o dispositivo for configurado pelo ABP, as chaves serão utilizadas até o dispositivo atualiza-las (Yang *et al.*, 2018).

Já o procedimento OTAA o dispositivo final envia uma solicitação de associação, que contém um número aleatório. Após o servidor receber a mensagem, ele deve aceitar ou não, caso o servidor não aceite o dispositivo não há mensagem de resposta. No caso de aceitação o servidor encaminhará uma mensagem de aceitação de associação ao dispositivo final, contendo um número gerado por ele. Após o recebimento da mensagem pelo dispositivo os dois lados usam os números para gerar as chaves *NwkSKey* e *AppSKey* (Yang *et al.*, 2018).

Os dispositivos finais utilizados neste trabalho foram ativados com o método ABP, pois como citado anteriormente ele possui parâmetros que possibilitam identificar um dispositivo. No método de ativação OTAA o *DevAddr* é um número aleatório e isso faz com que não se possa identificar um dispositivo por ele e por consequência o *firewall* proposto neste trabalho não se aplica a esses casos. A classe que o dispositivo implementa não interfere no funcionamento do *firewall*.

Esses protocolos possuem vulnerabilidades (Yang *et al.*, 2018). A documentação LoRaWAN v1.0.3 afirma que após aceitação do servidor ou redefinição do dispositivo final, os contadores de quadros de mensagem do dispositivo final e do servidor são redefinidos para zero (LORA ALLIANCE, 2021b). Tendo isso em vista, um dispositivo ativado por ABP usará novamente o valor do contador de quadros com 0. Sendo assim invasores podem capturar mensagens da sessão anterior e utilizá-las novamente, podendo interromper a comunicação de um dispositivo final com o servidor. Esse ataque pode ser realizado em ambos os métodos de ativação mas pode ser realizado com maior facilidade nos dispositivos ativados por ABP (Yang *et al.*, 2018).

Levando em conta que LoRaWAN é uma tecnologia relativamente nova, além da citada anteriormente, e pode existir vulnerabilidades que ainda nem foram descobertas. A seção a seguir aborda os trabalhos relacionados.

2.3 Trabalhos Relacionados

O LoRaWAN é uma relativamente tecnologia nova, sua primeira versão surgiu em 2015, desenvolvida pela lora Alliance, que motivou muitas pesquisas acadêmicas relacionadas ao desempenho do protocolo, mas poucas a respeito de questões relacionadas a segurança e vulnerabilidades. Como Yang *et al.* (2018) afirma em seu trabalho :

“Até onde sabemos, fomos os primeiros a estudar a segurança da pilha de protocolos LoRaWAN e suas vulnerabilidades de maneira sistemática.”

Sabendo da importância de uma análise aprofundada a respeito de segurança do protocolo LoRaWAN, o trabalho *Security Vulnerabilities in LoRaWAN* aborda o tema e ilustra como explorar vulnerabilidades encontradas no protocolo.

Yang *et al.* (2018) fornece uma visão geral a respeito a segurança do protocolo LoRa, também apresenta tipos de ataque que podem ser realizados em redes LoRa como:

- *Replay Attack*: Neste ataque um invasor pode capturar mensagens de sessões anteriores e reutiliza-las em outras sessões para interromper a comunicação entre o dispositivo final e o servidor.
- Espionagem: Baseando-se em mensagens capturadas anteriormente, cria-se um padrão para mensagem e derivar o texto criptografado.
- ACK *Spoofing*: O invasor possuindo um *gateway* malicioso pode redirecionar as mensagens de ACK para reconhecer outros quadros que não os originalmente recebidos pelo servidor.

O artigo conclui citando maneiras de mitigar os ataques e possíveis correções no protocolo para torna-lo mais seguro. Para evitar o *replay attack* o dispositivo final deve recodificar as chaves de segurança toda vez que o contador atingir o valor máximo. Se o dispositivo estiver usando o protocolo OTAA (LORA ALLIANCE, 2021b), ele deve passar novamente pelo protocolo quando o contador atingir o valor máximo, já se estiver ativado com protocolo ABP (LORA ALLIANCE, 2021b), deverá ser reconfigurado e as chaves alteradas. Já o ataque de espionagem pode ser mitigado se trocar o valor do contador por um número usado apenas uma vez, derivado de um gerador de números pseudo-aleatórios.

Além disso Yang *et al.* (2018) comenta que no desenvolvimento do trabalho uma nova versão do protocolo foi lançada e algumas vulnerabilidades foram corrigidas, porém como duas especificações do protocolo são suportadas reitera que os resultados encontrados ainda são válidos (Yang *et al.*, 2018).

2.4 Considerações Finais

Hoje em dia diversos setores estão adotando redes LoRa no seu dia a dia devido ao grande alcance de transmissão e conectividade. Um desses setores é a agricultura de precisão (IT MIDIA, 2021), que busca automatizar as áreas da pecuária e lavoura, para isso coleta-se a maior quantidade de dados possíveis através de sensores durante as operações rurais. Tendo em vista esse cenário a proteção desses dados é importante, pois a partir da interpretação dos mesmo são tomadas decisões referentes ao plantio, colheita e pulverização (IT MIDIA, 2021). Ataques a *gateways* que transmitem essas informações podem trazer desde desperdícios de insumos ou até prejuízos financeiros.

Sabendo que existem vulnerabilidades no protocolo LoRaWAN, sendo que algumas foram corrigidas, outras não e possivelmente ainda há vulnerabilidades que se quer foram encontradas *firewall*, localizado no *gateway*, que armazena informações sobre dispositivos finais e de seus pacotes como endereço, número do contador de pacotes, mensagens perdidas, mensagens enviadas pode prevenir comportamentos anormais e mitigar ataques contra este tipo de rede.

3 METODOLOGIA

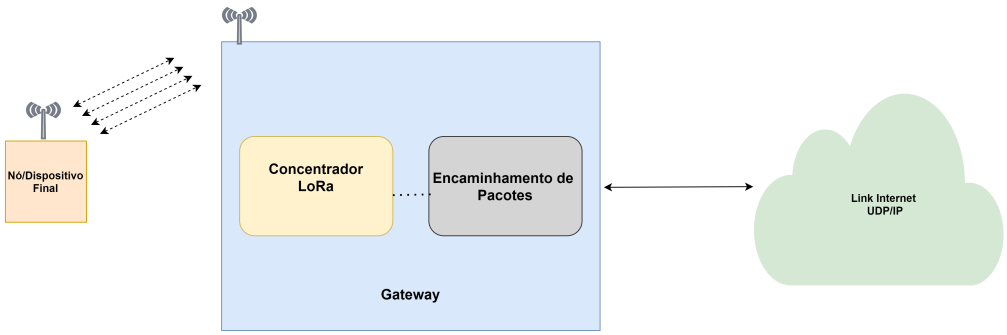
Este capítulo descreve os materiais e procedimentos usados para desenvolver um *firewall* para redes LoRaWAN, explicando sua arquitetura e apresentando seu comportamento no *gateway* LoRaWAN. Para um bom entendimento do *firewall* proposto, a Seção 3.1 demonstra o funcionamento e as características do *gateway* utilizado no presente trabalho. Já a Seção 3.2 explica como foi desenvolvido o *firewall* e ilustra seu funcionamento.

3.1 *Gateway* LoRa

Para alcançar o objetivo de construir o *firewall* para redes LoRaWAN, inicialmente foi necessário configurar um *gateway* e habilitá-lo para encaminhar pacotes aos servidores na Internet. O *gateway* utilizado neste trabalho é composto por concentrador LoRa de oito canais RisingHF 915MHz (Cytron Technologies, 2021) instalado em um *Raspberry Pi* 3 (Raspberry, 2021) e também pelo software de código aberto conhecido como RisingHF-Gateway (TECHNOLOGIES, 2021d), fornecido pela empresa Cytron Technologies (TECHNOLOGIES, 2021a).

Conforme a Figura 4 ilustra, o software é composto por duas partes: a primeira é o Concentrador LoRa, que é baseado em um receptor Semtech LoRa (TECHNOLOGIES, 2021b); Já a segunda, é o Encaminhamento de Pacotes (em inglês, Packet Forwarder) (TECHNOLOGIES, 2021c), que realiza a comunicação entre um concentrador LoRa e a Internet. Após instalado e devidamente configurado, o *gateway* LoRaWAN consegue encaminhar pacotes recebidos de dispositivos LoRa para a Internet.

Figura 4 – *Gateway* LoRa.



Fonte: Autoria própria (2022).

Na Figura 4, observa-se que o *gateway* recebe pacotes RF enviados pelos dispositivos através da camada física do LoRa (LORA ALLIANCE, 2021a), que são encaminhados para o Encaminhamento de Pacotes e convertidos para objetos *JavaScript Object Notation* (JSON). Esse objeto no formato JSON contém os pacotes, metadados e informações pertinentes ao *gateway*, que posteriormente são enviados através de um *link* UDP/IP para o servidor na Internet. O *gateway* também realiza o processo inverso, enviando pacotes de resposta da Internet para os dispositivos finais na rede LoRa.

Sabendo do funcionamento básico do *gateway* LoRa agora é possível entender como o *firewall* proposto foi implementado (ver Seção 3.2).

3.2 Desenvolvimento do *Firewall*

Com o intuito de construir um *firewall* para redes LoRaWAN primeiramente foi necessário conhecer o funcionamento do software RisingHF-Gateway, pois só assim seria possível constatar a possibilidade de construção do *firewall* e onde desenvolve-lo.

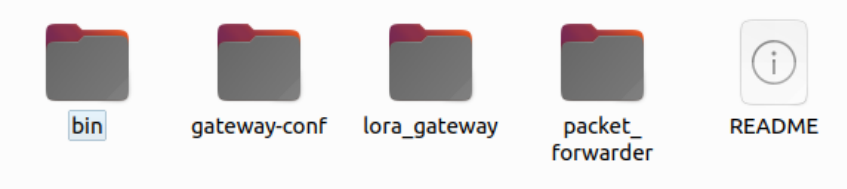
Então, há duas questões de pesquisa neste trabalho:

- Questão 1 - É possível criar um *firewall* para bloquear pacotes em redes LoRa?
- Questão 2 - Caso a resposta anterior seja “sim”, qual seria o melhor ponto do software do *Gateway* LoRa para desenvolver/implementar esse *firewall*, de forma que não influencie ou interfira o mínimo possível no desempenho do *gateway*?

Tais questões são respondidas no decorrer da pesquisa.

Para um melhor entendimento do *gateway* LoRa foi preciso explorar os arquivos e diretórios do código fonte do software RisingHF-Gateway (TECHNOLOGIES, 2021d). Na Figura 5 pode-se observar o diretório raiz do projeto RisingHF-Gateway. Através desta investigação foi possível entender como os diretórios estavam divididos, em qual arquivo o pacote enviado pelo dispositivo final era recebido e como esse pacote era convertido para objeto JSON.

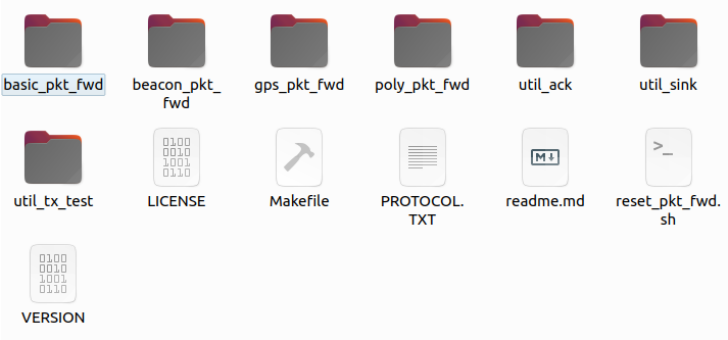
Figura 5 – Diretório raiz do *gateway* LoRaWAN



Fonte: Autoria própria (2022).

Durante a investigação do código, verificou-se que o diretório *Packet Forwarder* contém um arquivo chamado *PROTOCOL.TXT* (ver Figura 6) que traz uma breve descrição de cada arquivo presente no diretório. Segundo *PROTOCOL.TXT*, o arquivo responsável pela conversão dos pacotes vindos de dispositivos finais é o *poly_packet_forwarder.c*. A partir disso, uma análise do código presente no arquivo *poly_packet_forwarder.c* foi feita para identificar onde os pacotes são recebidos e se é possível bloqueá-los. Através desta análise concluiu-se que há uma estrutura que armazena as informações recebidas dos pacotes LoRa como, frequência, modulação, *timestamp* e outras informações pertinentes ao *gateway*. Sabendo disso foi necessário realizar outra pesquisa na documentação do desenvolvedor LoRaWAN (LORA ALLIANCE, 2021b), para verificar como estavam dispostas as informações nos quadros do pacote LoRa e quais informações o *gateway* tem acesso ao receber um pacote.

Figura 6 – Diretório *Packet Forward*



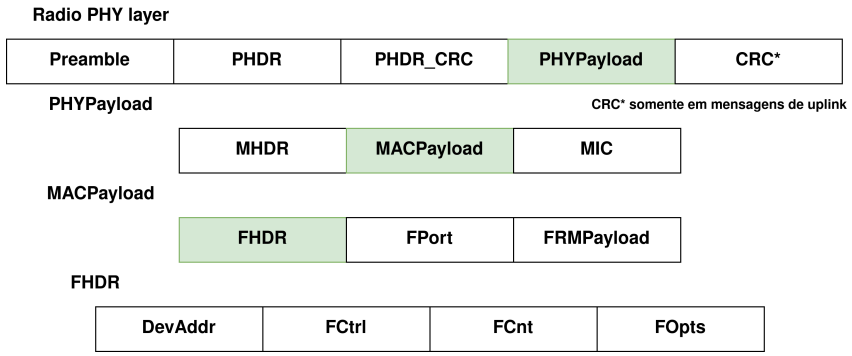
Fonte: Autoria própria (2022).

Após a investigação foi possível notar que toda mensagem LoRa leva consigo um *payload* de camada física (LORA ALLIANCE, 2021d), que inicia com um octeto de cabeçalho MAC (*MAC Header* (MHDR)), seguido por um quadro de carga útil (MAC *payload*) e terminado com um código (soma de verificação - *checksum*) de 4 octetos que realiza a verificação de integridade do pacote (LORA ALLIANCE, 2021d).

Já o quadro FHDR (Figura 7), localizado dentro do MAC *payload*, é composto pelo: endereço do dispositivo LoRa; um octeto de controle de quadro (chamado FCtrl) e dois octetos para o campo FCnt, que é responsável por contar os quadros e pode ter até quinze octetos para transportar comandos MAC (LORA ALLIANCE, 2021d).

Na Figura 7, observa-se que o *MACPayload* contém o cabeçalho do quadro (*Frame Header* (FHDR)) acompanhado pelo campo de porta FPort e do *FRMPPayload*. Após observar o formato do pacote LoRa, constatou-se que o campo *DevAddr* contém o endereço dos dispositivos LoRa, e portanto pode ser utilizado como filtro para pacotes recebidos dos dispositivos finais. Então, o campo *DevAddr* representa o endereço físico do dispositivo final e pode ser utilizado para criar mecanismos para liberar/bloquear dispositivos fundamentado nesta informação.

Figura 7 – Formato do pacote LoRa



Fonte: LoRa Alliance (2022).

Tendo em vista as informações presentes no pacote LoRa e sabendo o arquivo em que é realizada a conversão dos pacotes RF para objetos JSON, foi necessário localizar e averiguar no código de conversão o campo onde exatamente esta conversão acontecia para conseguir

verificar se realmente o campo *DevAddr* poderia ser utilizado na implementação de um *firewall* para redes LoRa. Após mais uma análise no arquivo *poly_packet_forwarder.c*, constatou-se que uma *thread* é responsável por interpretar os pacotes recebidos pelo *gateway*. Dentro dessa *thread*, há um laço de repetição responsável por iterar em um vetor que contém pacotes recebidos. Então esse foi identificado como sendo o lugar em que seria possível bloquear pacotes que se enquadrem em alguma regra de *firewall*, pois cada iteração deste laço de repetição representa o encaminhamento de um pacote. Assim é possível comparar o *DevAddr* do pacote que esta sendo encaminhando com as regras presentes no *firewall* e realizar ações de bloqueio/liberação caso necessário. Este trecho de código se encontra no arquivo *poly_packet_forwarder.c* a partir da linha 1.623 (TECHNOLOGIES, 2021c).

Depois de entender o funcionamento do código e a posição no código para implementar o *firewall* proposto, foram realizados pequenos testes de bloqueio como observado no trecho de Código 3.1. Nesse, a linha 1 representa o endereço de um dispositivo final que deve ser bloqueado, em seguida é iniciado o laço de repetição que itera sobre o pacote recebido onde são extraídas as informações do pacote para o encaminhamento (linha 2). A linha 4 foi inserida para o teste e tem como finalidade realizar a comparação do endereço presente no pacote recebido pelo *gateway* e a variável “deviceaddr”, esta comparação é realizada utilizando as posições iniciais do pacote onde encontra-se o *DeviceAddr*, caso haja correspondência entre os endereços a iteração do laço é interrompida o que representa o bloqueio do pacote (linha 6).

Após esse teste de bloqueio utilizando o endereço do dispositivo dentro do código (*hard-coded*), verificou-se que era viável utilizar o campo *DeviceAddr*, que contém o endereço do dispositivo final para bloquear pacotes. Foi possível definir então o local onde o *firewall* poderia ser implementado dentro do software e constatou-se que é possível implementar o *firewall* proposto. O que responde positivamente a primeira questão proposta no presente trabalho: É possível criar um *firewall* para bloquear pacotes em redes LoRa?. Ou seja, para essa pergunta a resposta é “sim”, é possível desenvolver um *firewall* para bloquear/liberar dispositivos em redes LoRaWAN.

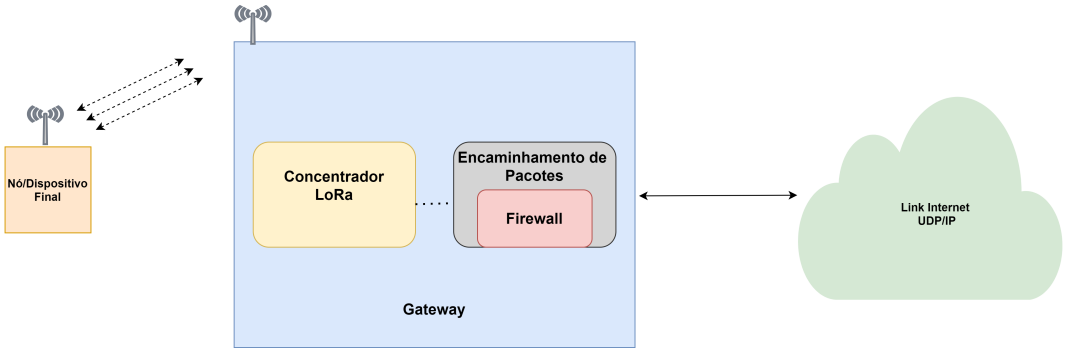
Assim de forma mais abstrata, a Figura 8 demonstra o local onde o *firewall* fica localizado após sua implementação dentro do *gateway*, que efetivamente está no arquivo *poly_packet_forwarder.c* do software LoRa, onde pode verificar o endereço do dispositivo que enviou o pacote antes do envio para a Internet.

```
1 uint8_t deviceaddr[4] = {0x26, 0x03, 0x1C, 0x2C};
2 for (i=0; i < nb_pkt; ++i) {
3   p = &rxpkt[i];
4   if(deviceaddr[0] == p->payload[4] && deviceaddr[1] == p->payload[3] &&
     deviceaddr[2] == p->payload[2] && deviceaddr[3] == p->payload[1]){
5       printf("bloqueado \n");
6       break;
```

```
7     }
8 }
```

Listing 3.1 – Teste de bloqueio de pacotes.

Figura 8 – Localização do *firewall* após o desenvolvimento.



Fonte: Autoria própria (2022).

3.2.1 Funcionamento do *Firewall*

Após identificar o local em que é possível realizar o bloqueio de pacotes, foi necessário modificar o código fonte do *gateway* para que as regras fossem inseridas conforme o necessário (não *hard-coded*) e para criar, alterar e excluir essas regras de forma mais amigável foi desenvolvida uma interface.

A interface proposta para o gerenciamento de regras recebe comandos fornecidos pelo usuário e os interpreta para criar, alterar ou excluir-las. E assim as salvar em um arquivo no formato JSON para que posteriormente o *gateway* tenha acesso a essas informações. Com isso ele pode interpretar e aplicar as regras aos pacotes recebidos.

Para o desenvolvimento do *firewall* para redes LoRa foi necessário elaborar regras com a intenção de permitir ao administrador filtrar pacotes recebidos pelo *gateway*. Inicialmente foram criadas duas ações, *Deny* e *Allow*, onde *Deny* representa bloqueio e *Allow* liberação. Com a intenção de armazenar conjuntos de endereços em que as ações de bloqueio/liberação serão aplicadas foram criadas a *DenyList* e *AllowList*. A *AllowList* guarda os endereços de dispositivos que serão liberados pelo *firewall*. Já na *DenyList* ficam os endereços proibidos de utilizar o *gateway*.

Estas ações foram criadas com o objetivo de proporcionar segurança para o *gateway*, pois há possibilidade de que somente dispositivos conhecidos (que estão presentes na *AllowList*) trafegarem pelo *gateway* ou para que o tempo de processamento do *gateway* não seja desperdiçado com dispositivos indesejados, possibilitando ao administrador melhor controle de fluxo da sua rede. Em trabalhos futuros outras ações podem ser implementadas com o in-

tuito de possibilitar outras formas de bloqueio, como bloquear/liberar somente mensagens de *Uplink*/*Downlink*.

O *firewall* para redes LoRaWAN opera da seguinte forma:

- Utilizando a interface, endereços de dispositivos poderão ser adicionados a dois tipos de lista, *DenyList* e *AllowList*.
- As listas são armazenadas em um arquivo de configuração.
- Quando *gateway* é iniciado, o conteúdo do arquivo é interpretado e são criadas as regras para a *Deny* e *Allow* com a finalidade de serem consultadas posteriormente. Pacotes que são originados/destinados por endereços de dispositivos que combinem com algum endereço contido na *AllowList* serão encaminhados normalmente, já os que estiverem endereços na *DenyList* serão descartados/bloqueados.
- Assim quando um pacote é recebido, uma checagem é realizada na quantidade de endereços presentes na *AllowList*, caso exista o endereço da lista é comparado com o campo de endereço do pacote (*DevAddr*) e se houver correspondência o pacote é imediatamente liberado. Se não existir itens na *AllowList* ou não houver nenhuma correspondência com os endereços presentes nela, uma checagem na quantidade de itens presentes na *DenyList* é realizada. Caso exista endereços, ocorre uma nova comparação entre o *DevAddr* do pacote com os endereços da *DenyList* e se houver correspondência o pacote é imediatamente descartado. Não havendo correspondência em nenhuma das duas listas o encaminhamento retoma seu funcionamento padrão.

A Figura 9 ilustra o fluxo de processamento do *firewall*.

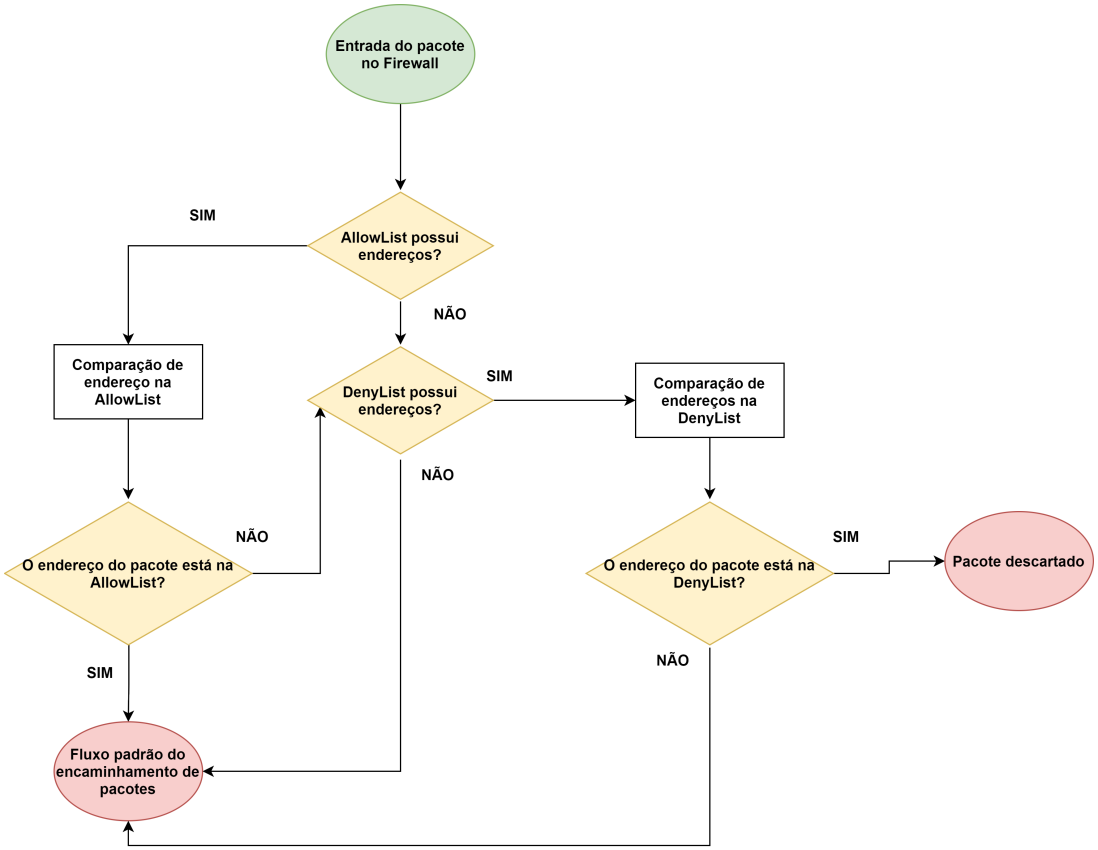
Depois de definido o funcionamento do *firewall* e suas ações, foi necessário criar uma forma do usuário inserir regras no *firewall*, pensando nisso foi criada uma interface que possibilita, através de comandos, criar e excluir regras. Uma explicação mais detalhada é apresentada na Seção 3.2.2.

3.2.2 Interface

Neste trabalho, o termo interface refere-se a um *script* que executa um interpretador de linha de comando. Esse *script* foi desenvolvido com a linguagem *Python* (FOUNDATION, 2021) e tem como objetivo principal gravar regras, geradas através de comandos fornecidos pelo usuário, em um arquivo no formato JSON. O arquivo é salvo neste formato, pois o *gateway* utiliza uma biblioteca própria com funções para extrair informações de arquivos deste formato. A Figura 10 apresenta um exemplo da interface.

A principio a interface aceita comandos para adicionar dispositivos a *DenyList*, *AllowList* e permite excluir regras criadas. Além disso há um comando que mostra novamente a lista de

Figura 9 – Funcionamento do *firewall* para redes LoRa.



Fonte: Autoria própria (2022).

Figura 10 – Interface para o inserção de regras no *firewall*

```
Firewall for Gateways LoRaWAN

    Adiciona dispositvos a allowList:  add *deviceAddress* deny
    Adiciona dispositvos a denyallow:  add *deviceAddress* allow

    Remover regra:  remove *deviceAddress*
    Sair do programa: exit
    Exibir comandos novamente: help
```

Fonte: Autoria própria (2022).

comandos aceitos pela interface, a Tabela 1 ilustra a sintaxe dos comandos aceitos pelo *firewall*. Os comandos precisam ter o seguinte formato: comando (*add/remove*), endereço do dispositivo *DeviceAddress* (se preenchido como zero representa todos os endereços) e ação (*allow/deny*), comandos diferentes desta sintaxe não serão aceitos. No caso do comando *remove* só é necessário o endereço do dispositivo. Tanto ações de *allow* quanto *deny* são armazenadas no mesmo arquivo e somente são separadas quando o *gateway* é iniciado. Cada conjunto de regra, que é composto por ação e endereço, são armazenados em suas respectivas listas *DenyList* e *AllowList*.

O trecho de Código 3.2, é um exemplo de arquivo que contém regras gerado pela interface, nele observa-se um conjunto de dados (objetos) que representa as regras do *firewall*

chamado *firewall_conf* o qual possui um vetor de objetos chamado *nodes*, onde cada posição armazena o endereço e a ação que será aplicada ao dispositivo.

```
{
        "firewall_conf": {
                "nodes": [
                        {
                                "addr": "204309",
                                "rule": "deny"
                        },
                        {
                                "addr": "309808",
                                "rule": "allow"
                        }
                ]
        }
}
```

Listing 3.2 – Exemplo de arquivo de configuração do *firewall*.

Tabela 1 – Comandos *Firewall*

*add *deviceAddress* allow*	Adiciona dispositivos a *Allow List*
*add *deviceAddress* deny*	Adiciona dispositivos a *Deny List*
*remove *deviceAddress**	Remove todas as regras que possuem aquele endereço
help	Exibe os comandos novamente
exit	Encerra a interface

Ilustrando a utilização do *firewall* para redes LoRaWAN, imagine o seguinte exemplo: "Um dado usuário possui um *gateway* LoRaWAN em sua fazenda e instalou um dispositivo LoRa para monitorar a temperatura de sua plantação. Após algum tempo de uso notou que as atualizações das informações fornecidas pelo dispositivo estavam demorando muito. Preocupado com essa situação, foi ao console do seu servidor para verificar o que estava gerando este problema e percebeu que diversos outros dispositivos desconhecidos estavam trafegando pelo seu *gateway*. Visando melhorar o tempo de atualização de suas informações o usuário/administrador da rede LoRaWAN adicionou o endereço de seu dispositivo na *AllowList* e adicionou o endereço zero na *DenyList* para que somente seu dispositivo fosse retransmitido pelo *gateway*. O comando utilizado para adicionar um dispositivo na *AllowList* é apresentado na Figura 11, assim o comando "add" é seguido pelo endereço do dispositivo e pela ação *allow*".

Figura 11 – Exemplo de comando para adicionar um endereço a *AllowList*

add	260311B0	allow
Comando	Endereço	Ação

Fonte: Autoria própria (2022).

O exemplo citado anteriormente demonstra uma de várias formas em que as regras podem ser utilizadas para auxiliar na segurança e desempenho de redes LoRaWAN.

3.2.3 Aplicando as regra no *Gateway* LoRa

Após a definição da estrutura das regras (comando, endereço e ação) do *firewall* proposto para redes LoRaWAN, foi preciso alterar o código fonte do *gateway* com a intenção de que as regras registradas no arquivo fossem aplicadas aos pacotes recebidos. Para isso, foi necessário recuperar as regras contidas no arquivo do *firewall* e armazená-las em suas respectivas listas (*DenyList/AllowList*).

Com a intenção de contribuir com as boas práticas de programação e manter o código limpo, foram utilizados métodos e estruturas já presentes no código fonte do *gateway*. Sendo assim, para a serialização do arquivo JSON utilizou-se a biblioteca de código aberto *Parson.h* (Krzysztof Gabis, 2021), que possibilita interpretar objetos JSON e converte-los em variáveis na linguagem C (GNU, 2021).

Para o funcionamento básico do *firewall*, foram criadas variáveis para armazenar informações como as listas, endereços e variáveis de controle, como observado no trecho de Código 3.3. Nas linhas 1 e 2 são definidos os contadores da *DenyList* e *AllowList* e nas linhas 3 e 4 as listas são criadas. Também foi criada a função *parse_node_for_firewall*() que é responsável por preencher as variáveis descritas anteriormente (Código 3.3), utilizando métodos presentes na biblioteca *Parson.h*, com as informações extraídas do arquivo.

```
1 static uint8_t denny_list_count = 0;
2 static uint8_t allow_list_count = 0;
3 static uint32_t denny_list_addr[MAX_NODES];
4 static uint32_t allow_list_addr[MAX_NODES];
```

Listing 3.3 – Variáveis para controle do *firewall*.

Depois das variáveis preenchidas foi preciso analisar o endereço do pacote recebido para aplicar as regras do *firewall* e verificar a presença dele na *DenyList* ou *AllowList*. Para isso é preciso obter o endereço do pacote, sabendo que ele se faz presente nas primeiras posições do pacote e que tem-se o ponteiro que possui a referência do pacote, constatou-se que ali seria o lugar que o endereço poderia ser obtido.

O Algoritmo 1 é uma representação da implementação do *firewall*, que recebe como entrada o cabeçalho do pacote LoRa e retorna um *boolean* representando a liberação/bloqueado do pacote. Primeiramente o endereço do dispositivo é convertido, isso é necessário para comparação com os endereços contidos nas listas (*DenyList*/*AllowList*), assim é atribuído a uma variável (linha 5). Após isso é verificado a quantidade de endereços presentes na *AllowList* (linha 6) e caso exista endereços (linha 7), um laço de repetição é iniciado e cada posição da lista é comparada com o endereço do pacote a fim de encontrar correspondência (linhas 8 e 9). Se houver correspondência ou o endereço constar na lista seja 0 (o que representa que todos os endereços estão permitidos), é atribuído verdadeiro a uma variável de controle e o laço é encerrado (linhas 10 e 11). Se não encontrar nenhuma correspondência do endereço na *AllowList* e a *DenyList* possuir endereços (linha 16), um novo laço de repetição é iniciado (linha 17), iterando sobre a *DenyList* com a intenção de verificar a presença do endereço do dispositivo na lista, caso encontre ou o endereço presente na lista seja 0 (o que representa que todos os endereços estão bloqueados) é atribuído *false* a variável de controle e o laço é encerrado (linhas 18 a 20). Se o laço chegou ao fim e o endereço não estava presente na *DenyList* a variável de controle recebe verdadeiro (linha 25). Após essas verificações a variável de controle é retornada e se for verdadeira o pacote encaminhado normalmente, já se for falsa ele é descartado.

3.3 Considerações Finais

Com a solução proposta foi possível desenvolver um *firewall* para redes LoRa, que bloqueia pacotes que não estão de acordo com a regra vigente. Assim respondendo as perguntas levantadas no inicio do trabalho, pois com esta solução é possível filtrar os pacotes que irão ser retransmitidos pelo *gateway*. Pensando em trabalhos futuros pretende-se implementar novas políticas no *firewall*, como bloquear tipos de mensagens *Uplink*/*Downlink*, para que assim ele tenha tenha mais opções.

O Capítulo 4 apresenta os resultados da implementação e testes de desempenho realizados no *gateway*.

Algoritmo 1 – Algoritmo do *firewall* para redes LoRaWAN

```
1: INPUT: endereco_pacote_recebido
2: OUTPUT: libera_pacote
3: i ← 0
4: libera_pacote ← false
5: endereco_convertido ← converte_endereco(endereco_pacote_recebido)
6: quantidade_allowList ← obtem_quantidade_allowList()
7: se (quantidade_allowList > 0) então
8:    para (i < quantidade_allowList) faça
9:       se endereco_convertido == allowList[i] || allowList[i] == 0 então
10:         libera_pacote ← true
11:         break
12:      finaliza se
13:   finaliza para
14: senão,
15:   quantidade_dennyList ← obtem_quantidade_denyList()
16:   i ← 0
17:   se (quantidade_dennyList > 0 && !libera_pacote) então
18:      para (i < quantidade_denyList) faça
19:         se endereco_convertido == denyList[i] || denyList[i] == 0 então
20:            libera_pacote ← false
21:            break
22:         finaliza se
23:      finaliza para
24:   finaliza se
25:   libera_pacote ← true
26: finaliza se
27: retorna libera_pacote
```

4 EXPERIMENTOS E RESULTADOS

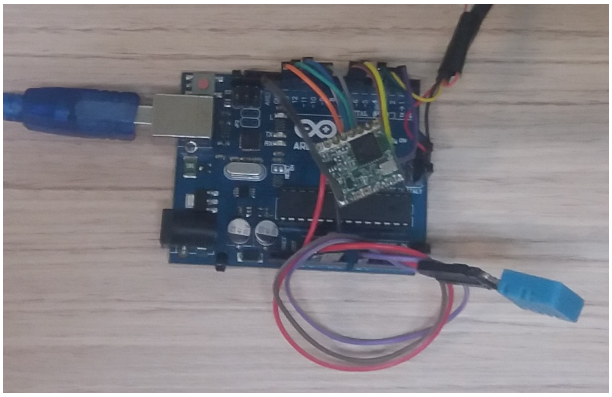
Após o desenvolvimento do *firewall* para redes LoRaWAN, foi preciso verificar o seu funcionamento e desempenho em ambiente real, para isso foram elaborados diferentes experimentos para diferentes cenários de testes com e sem o uso do *firewall*.

4.1 Experimentos

Localizado na Universidade Tecnológica Federal do Paraná (UTFPR) campus Campo Mourão, o cenário de testes é composto por um módulo LoRa RFM95W (Hoperf, 2021) instalado em um Arduíno Uno (ARDUINO, 2021) ilustrado Figura 12 ilustra. Juntamente com um *gateway* LoRa de oito canais *RisingHF* 915 MHz instalado em um *Raspberry Pi* como é possível observar na Figura 13 (Raspberry, 2021). Ambos estão hospedados na plataforma The Things Network, uma plataforma descentralizada de código aberto para IoT, que pode hospedar servidores, aplicações, dispositivos finais e *gateways* (The Things Network, 2021).

A Figura 14 representa o cenário de testes proposto para os experimentos. Todos os experimentos foram realizados com mensagens reais de *UpLink* enviadas pelo dispositivo final. O cenário utilizado nos experimentos conta somente com um único dispositivo final para testes, o que pode limitar a cobertura dos experimentos.

Figura 12 – Dispositivo final utilizado no cenário de testes.



Fonte: Autoria própria (2022).

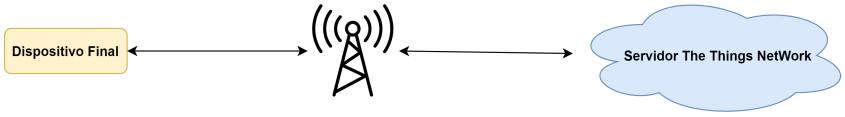
Após o cenário devidamente configurado e funcionando corretamente foi adicionado o endereço do dispositivo, contido no cenário de testes, na *DenyList*. Com a intenção de verificar se o *firewall* iria bloquear o pacote enviado pelo dispositivo final.

A Figura 15 demonstra o resultado da inserção de uma nova regra no arquivo do *firewall*. Ela contém o endereço do dispositivo utilizado no experimento seguido pela ação. Após a regra ser inserida no arquivo, o *gateway* foi iniciado e os pacotes do dispositivo final eram descartados, mostrando que o *firewall* para redes LoRaWAN consegue bloquear o encaminhamento de pacotes de dispositivos através de seus endereços.

Figura 13 – Gateway utilizado no cenário de testes.

Fonte: Autoria própria (2022).

Figura 14 – Cenário de testes.



Fonte: Autoria própria (2022).

Figura 15 – Arquivo de configuração do *firewall*

```
{
        "firewall_conf": {
                "nodes":
                [ { "addr": "260311B0",
                    "rule":  "deny"}]
        }

}
```

Fonte: Autoria própria (2022).

Sabendo que o *firewall* funciona corretamente, foi preciso verificar se a implementação do *firewall* prejudica o desempenho do *gateway* LoRaWAN. Sendo assim, foi necessário elaborar experimentos para realizar essa análise. A função *clock()* (CPLUSPLUS, 2021) que pertence a biblioteca *time.h* foi usada para verificar o tempo de execução. Esta função retorna o números de *clocks* desde que o programa foi iniciado. Dividindo esse tempo pela constante *CLOCKS_-*

PER_SEC é possível obter o número de segundos usados pela *Central Processing Unit* (CPU). Para fins estatísticos todos os experimentos foram realizados trinta vezes.

Para realização dos experimentos, no inicio da execução do encaminhamento de pacotes, foi inserida uma variável chamada *begin* que guarda o tempo inicial e ao final da execução a variável *end*, para guardar o tempo final. Esses tempos foram subtraídos e o resultado dividido pela constante *CLOCKS_PER_SEC*, como é possível observar na Figura 16.

Figura 16 – Modificações para obter o tempo de execução do código

```
clock_t begin = clock();

/*Execução*/

clock_t end = clock();
double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
```

Fonte: Autoria própria (2022).

Sabendo de como foi criado e quais eram os componentes do cenário de testes, as seções seguintes tem como intuito descrever cada experimento realizado neste trabalho e posteriormente demonstrar os resultados obtidos com eles.

4.1.1 Experimento 1 - Base

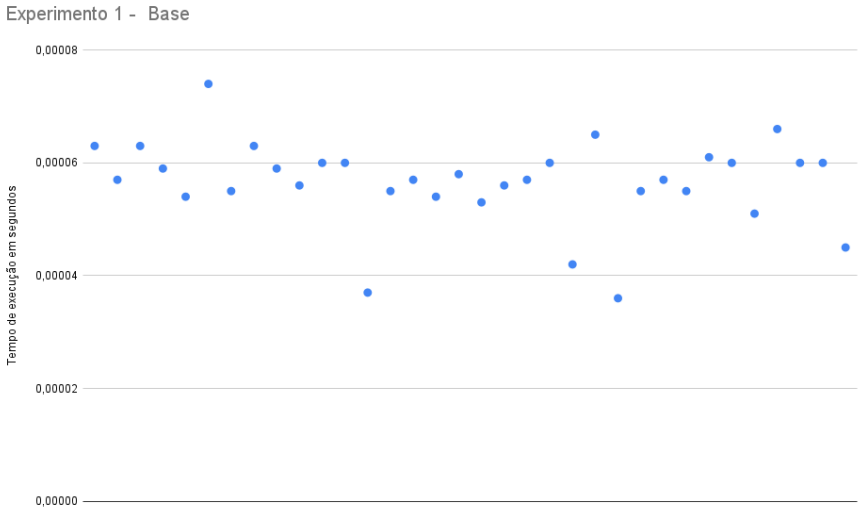
Este experimento consistiu em analisar o tempo de execução do encaminhamento de pacotes sem a implementação do *firewall*, a fim de obter o tempo que o *gateway* leva para normalmente realizar está ação. A Figura 17 trás o gráfico dos tempos obtidos na execução dos testes, observa-se que o tempo varia entre 0.000036 e 0.000074 segundos, com valor médio de 0.000057 segundos e desvio padrão de 0.000007612554131. Através deste experimento obteve-se a base para análise dos experimentos posteriores.

4.1.2 Experimento 2 - Inicial

Após verificar o tempo que o encaminhamento leva normalmente para encaminhar pacotes, o próximo experimento realizado foi verificar o tempo de execução com o *firewall* implementado, porém com as listas *DenyList* e *AllowList* vazias. Este experimento busca verificar se a estrutura criada para desenvolver o *firewall* influencia no tempo de execução do encaminhamento de pacotes.

A Figura 18 ilustra os resultados obtidos pelo experimento 2, os tempos variaram entre 0.000047 e 0.000074 segundos com o tempo médio de 0.000058 segundos e desvio padrão de 0.000006893963879. É possível observar na Tabela 2 a comparação entre o Experimento 1 e o Experimento 2, nota-se que houve um aumento de apenas 0.000001 segundo no tempo que o

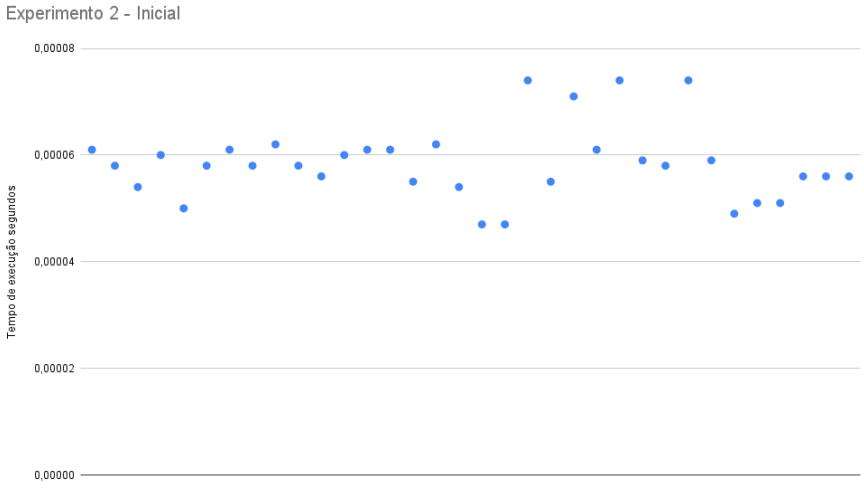
Figura 17 – Resultados do experimento 1 - Base



Fonte: Autoria própria (2022).

encaminhamento leva para encaminhar o pacote, o que mostra que as estruturas criadas para o desenvolvimento do *firewall* não afetaram o tempo de encaminhamento de pacotes.

Figura 18 – Resultados do experimento 2 - Inicial



Fonte: Autoria própria (2022).

Tabela 2 – Comparação entre experimento 1 e 2

Experimento	Média	Desvio Padrão
1	0.000057	0.000007612554131
2	0.000058	0.000006893963879
Diferença	0.000001	0.000000718590252

4.1.3 Experimento 3

O experimento 3 tem como intuito analisar o tempo de execução do encaminhamento de pacotes quando existem regras de bloqueio/liberação pertencentes ao *firewall*, fazendo com que a *DenyList* e *AllowList* possuam endereços, o que resulta em buscas a fim de encontrar alguma correspondência.

Este experimento foi dividido em duas partes, primeiramente foi inserido um endereço na *AllowList*, sendo ele o endereço do dispositivo utilizado no cenário de testes e um endereço desconhecido (endereço de um dispositivo que não existe - ou seja, não haverá correspondência para esse endereço durante o teste) na *DenyList*. Já na segunda parte do experimento foi realizado o processo contrário, inserindo o endereço do dispositivo conhecido na *DenyList* e o endereço desconhecido na *AllowList*.

A primeira parte do experimento obteve o resultado médio de 0.000063 segundos com desvio padrão de 0.000009242427165, já a segunda parte resultou no tempo médio de 0.000064 segundos com o desvio padrão de 0.000008882955395.

A Tabela 3 apresenta a média e o desvio padrão dos experimentos 1, 2 e 3, observando estes resultados e comparando-os com os experimentos anteriores, tem-se um aumento de aproximadamente 0.000006 segundos para a primeira parte e de aproximadamente 0.000007 segundos para a segunda parte. O que é um aumento relativamente pequeno tendo em vista o beneficio que o *firewall* trás com relação a segurança do *gateway*.

Tabela 3 – Comparação entre Experimento 1, 2 e 3

Experimento	Média	Desvio Padrão
1	0.000057	0.000007612554131
2	0.000058	0.000006893963879
3.1	0.000063	0.000009242427165
3.2	0.000064	0.000008882955395

4.1.4 Experimento 4

Para verificar se a quantidade de endereços presentes na *DenyList* e *AllowList* do *firewall* influenciam no desempenho do encaminhamento de pacotes, foi criado o seguinte cenário: As duas listas (*DenyList* e *AllowList*), foram preenchidas variando a quantidade de endereços, com a intenção de que fosse realizada a checagem nas duas listas por completo e não houvesse nenhuma correspondência. Os testes foram realizados com 10, 100, 1.000, 5.000, 10.000 e 50.000 endereços.

A Tabela 4 apresenta a média e o desvio padrão obtidos com o experimento 4, analisando os resultados pode-se observar que as quantidades de endereços 10, 100 e 1.000 são semelhantes a os adquiridos nos experimentos anteriores, porém observa-se que quando as duas listas possuem quantidades acima de 5.000 endereços, o tempo que o pacote leva para

ser encaminhado aumenta gradativamente. Levando em conta a quantidade de endereços em cada lista o tempo é aceitável.

Tabela 4 – Média dos resultados obtidos com Experimento 4

Endereços na *DenyList* e *AllowList*	**Média**	**Desvio Padrão**
10	0,0000577	0,000005495661821
100	0,0000647	0,000009559793678
1.000	0,0000699	0,000004253598882
5.000	0,0001094	0,00001699631468
10.000	0,0001585	0,000009027735043
50.000	0,0005717	0,00001788214932

4.1.5 Experimento 5

Este experimento foi dividido em duas partes. A primeira parte consiste em preencher as duas listas variando a quantidade de endereços e preenchendo a última posição da *DenyList* com o endereço do dispositivo final utilizado no cenário de teste. Já a segunda parte do experimento foi idêntica a primeira, porém a lista que continha o endereço do dispositivo final era a *AllowList*. Os testes foram realizado variando a quantidade de endereços em 10, 100, 1.000, 5.000, 10.000 e 50.000.

A Tabela 5 apresenta a média e o desvio padrão obtido na primeira parte do experimento 5. Nota-se que quando as listas possuem até 1.000 endereços os resultados são semelhantes,já quando os endereços ultrapassam a marca de 5.000 o tempo começa a aumentar, chegando a média de 0.0005905 segundos quando existem 50.000 endereços.

Tabela 5 – Média dos resultados obtidos com o Experimento 5.1

Endereços *DenyList*	**Média**	**Desvio Padrão**
10	0.0000608	0.000004771201926
100	0.0000645	0.00001073312629
1.000	0.0000687	0.000008042702125
5.000	0.0001084	0.00001627055729
10.000	0.0001616	0.000007623014156
50.000	0.0005905	0.00001627055729

A Tabela 6 trás a média e o desvio padrão obtido na segunda parte do experimento 5. É possível observar que as médias quando ultrapassam 5.000 endereços também aumentam, mas em uma proporção menor comparada a primeira parte do experimento. Isso acontece porque como o endereço do dispositivo final encontra-se na ultima posição da *AllowList*, não há necessidade de percorrer a *DenyList* poupando assim tempo de execução da CPU.

Com a realização destes experimentos foi possível verificar se o desenvolvimento do *firewall* afeta o desempenho do *gateway* LoRaWAN. Os resultados serão apresentados na Seção 4.2.

Tabela 6 – Média dos resultados obtidos com o Experimento 5.2

Endereços *AllowList*	Média	Desvio Padrão
10	0.0000557	0.000006931603454
100	0.0000648	0.00001073312629
1.000	0.0000683	0.000007470170565
5.000	0.0000863	0.00001317385952
10.000	0.0001083	0.000007834201468
50.000	0,0003114	0,00001198773319

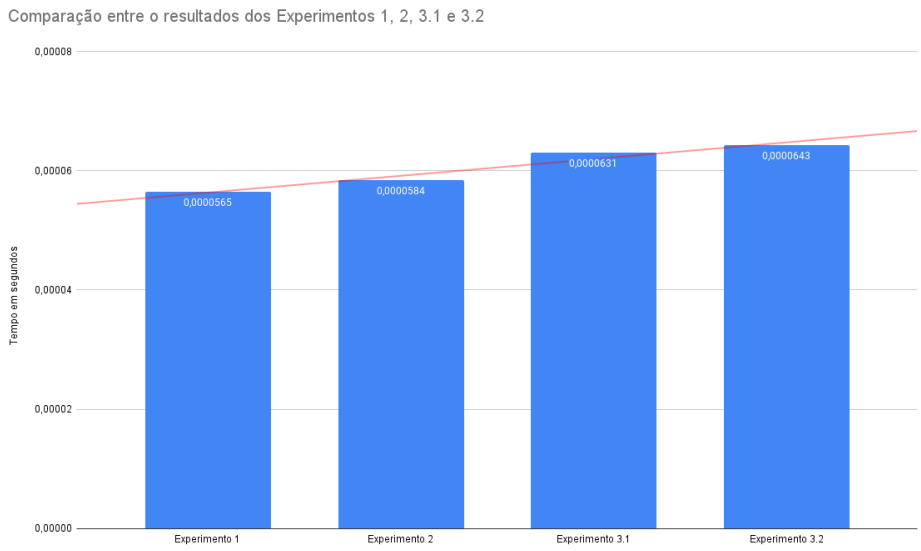
4.2 Resultados

Esta seção analisa os resultados alcançados com o desenvolvimento do *firewall* para redes LoRaWAN, atestando que é possível bloquear pacotes de dispositivos finais através do seu endereço e que esta solução não acarreta problemas de desempenho no *gateway* LoRaWAN. Como descrito na Seção 4.1 foram realizados vários experimentos para verificar o tempo de execução do encaminhamento de pacotes em diversos cenários.

O primeiro teste realizado foi denominado experimento base, pois ele verifica o tempo que o encaminhamento de pacotes leva para encaminhar um pacote sem nenhuma implementação do *firewall*. O resultado obtido foi que em média o pacote leva 0.000056 segundos para ser encaminhado, com desvio padrão de 0.0000076.

Sabendo do tempo de execução do encaminhamento de pacotes sem o *firewall*, foi possível comparar o tempo de execução do experimento 1 com os resultados obtidos no experimento 2 e 3, como é possível observar na Figura 19.

Figura 19 – Gráfico de comparação entre os resultados dos experimentos 1, 2, 3.1 e 3.2



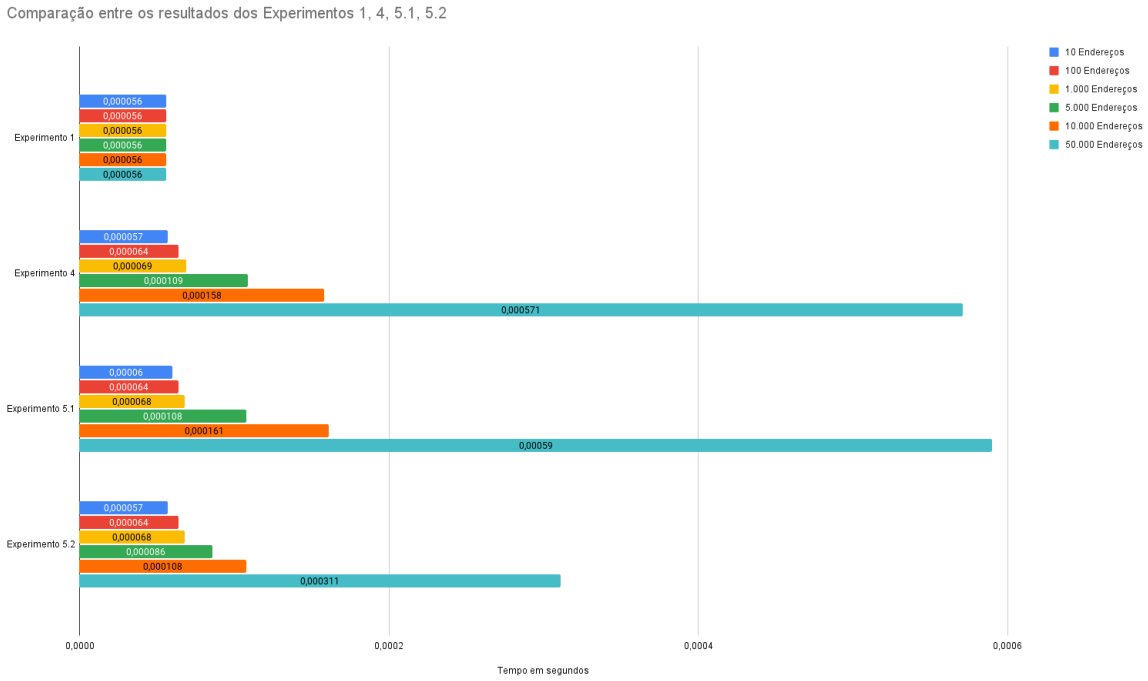
Fonte: Autoria própria (2022).

Tendo em vista os dados apresentados na Figura 19, é possível observar que a diferença entre o Experimento 1 e 2 foi de aproximadamente 0.0000019 segundos. Comparando o Expe-

rimento 1 com o Experimento 3.1 e 3.2 obteve-se a diferença de aproximadamente 0.0000066 e 0.0000078 segundos respectivamente. Logo conclui-se que a implementação do *firewall* com a *DenyList* e *AllowList* vazias ou com um endereço em cada não afeta o desempenho médio do encaminhamento de pacotes.

Após realizar as primeiras análises, foi preciso verificar os experimentos que variavam a quantidade de endereços na *DenyList* e *AllowList*. Os experimentos 4, 5.1 e 5.2 variaram a quantidade de endereços presentes nas listas entre 10, 100, 1.000, 5.000, 10.000 e 50.000 endereços com o intuito de verificar qual o quantidade de endereços que iria representar uma queda significativa no desempenho do encaminhamento de pacotes. A Figura 20 apresenta uma comparação entre os resultados experimentos 1, 4, 5.1 e 5.2.

Figura 20 – Comparação entre os resultados dos experimento 1, 4, 5.1 e 5.2



Fonte: Autoria própria (2022).

Observando a Figura 20 conclui-se que os tempos de execução aumentam de acordo com a quantidade de endereços na lista, o que é normal pois quanto maior a quantidade de endereços mais comparações são necessárias para verificar se o endereço do pacote está na *DenyList* ou *AllowList*. Também nota-se que os tempos do experimento 5.2 com uma quantidade de endereços maior que 5.000 são menores comparados com os experimentos 4 e 5.1. Isso ocorre porque quando há uma comparação positiva entre o endereço do pacote com um endereço na *AllowList* o pacote é liberado e não há necessidade de percorrer a *DenyList*, o que demonstra que a política de bloquear todos os endereços e liberar somente os que estão na *AllowList* é a que menos impacta no desempenho do encaminhamento de pacotes.

Pensando em um ambiente de aplicação real em que um administrador pode utilizar o *firewall* para liberar/bloquear endereços, dificilmente as duas listas (*DenyList* e *AllowList*) estariam simultaneamente com a quantidade máxima de endereços que trafegam pelo *gateway*, pois é possível bloquear/liberar todos os endereços somente com uma regra, logo mesmo com uma quantidade elevada de dispositivos finais trafegando pelo *gateway* ainda teríamos uma quantidade pequena de endereços presentes nas listas. Então pode-se afirmar que a implementação do *firewall* para redes LoRaWAN é possível e viável pois o seu desenvolvimento não atrapalha o funcionamento e desempenho normal do *gateway*.

4.3 Considerações Finais

Tendo em vista os resultados apresentados, é possível concluir que a técnica utilizada para o construção do *firewall* para redes LoRaWAN é eficaz e pode vir a ser implementada em *gateways* LoRa que possuem configurações semelhantes ao utilizado no cenário de teste, a fim de mitigar ataques ao mesmo.

5 CONCLUSÕES

Nesta monografia foi desenvolvido e testado um *firewall* para redes LoRaWAN. O *firewall* mostrou-se eficiente pois bloqueou e liberou pacotes de dispositivos finais que trafegam por um *gateway* LoRaWAN e seu desenvolvimento não afetou desempenho do *software*.

O desempenho do *firewall* foi analisado por meio de experimentos. Para a realização destes experimentos foi criado um cenário de testes contendo um dispositivo final e um *gateway* LoRaWAN conectados a um servidor na Internet. Os experimentos tiveram como base a ideia de verificar se a implementação do *firewall* afetava o desempenho normal do software e se a quantidade de endereços existentes nas listas de bloqueio/liberação influenciavam no tempo que o pacote leva para ser encaminhado.

Os resultados dos experimentos mostraram que as estruturas criadas para a implementação do *firewall* para redes LoRaWAN não influenciaram no desempenho do *software*. Também notou-se que quanto maior a quantidade de endereços existentes nas listas, maior o tempo para se encaminhar o pacote. Levando em conta que é possível combinar ações para bloquear/liberar pacotes, a quantidade de endereços existentes nas listas (*DenyList* e *AllowList*) do *firewall* irá ser pequena, o que não afetará o desempenho do encaminhamento de pacotes.

Ao longo deste trabalho foram levantadas duas questões principais. A primeira questão, é se havia possibilidade de desenvolver o *firewall* para bloquear pacotes de rede LoRa. A partir dos experimentos realizados conclui-se que é possível desenvolver um *firewall* para bloquear dispositivos LoRa possuindo seu endereço e que foram ativados por ABP. Já a segunda questão trata, caso fosse possível desenvolver o *firewall*, qual seria o melhor ponto do software do *gateway* LoRa para desenvolver/implementar este *firewall*, de forma que não influencie e interfira o mínimo possível no desempenho do *gateway*. Por meio de testes realizados, constatou-se que o *firewall*, no local onde foi desenvolvido (encaminhamento de pacotes), não atrapalha o funcionamento normal do *software*. A resposta da segunda questão também foi afirmativa.

A solução proposta neste trabalho pode auxiliar na proteção que falta em *gateways* LoRa. Pois bloqueando o tráfego de pacotes de dispositivos desconhecidos no *gateway* é possível minimizar riscos de ataques ao *gateway* e proteger os dados que trafegam no mesmo. Além disso coletar informações dos pacotes LoRa pode prevenir e mitigar ataques cibercriminosos.

Para trabalhos futuros propõe-se criar políticas e novas ações para o *firewall* com o intuito de torna-lo mais amplo e assim aumentar a gama de regras presentes no *firewall*. Também pretende-se estudar a sinergia entre *blockchain* (IBM, 2021) e redes LoRaWAN.

REFERÊNCIAS

ALLIANCE, L. **What is the LoRaWAN Specifications?** 2021. Disponível em: https://www.lora-alliance.org/. Acesso em: 26 jul. 2021.

ARDUINO. **What is Arduino?** 2021. Disponível em: https://www.arduino.cc/en/Guide/Introduction. Acesso em: 20 set. 2021.

ASHTON, K. That "internet of things". **RFiD Journal**, 2009.

Atlas VPN. **Cybercrime cost the world over $1 trillion in 2020**. 2021. Disponível em: https://atlasvpn.com/blog/cybercrime-cost-the-world-over-1-trillion-usd-in-2020. Acesso em: 1 ago. 2021.

Atlas VPN. **IoT malware attacks worldwide surge by 66% to over 50 million in 2020**. 2021. Disponível em: https://atlasvpn.com/blog/iot-malware-attacks-worldwide-surge-by-66-to-over-50-million-in-2020. Acesso em: 1 ago. 2021.

AUGUSTIN, A. *et al.* A study of lora: Long range & low power networks for the internet of things. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 16, n. 9, p. 1466, 2016.

Avast. **O guia essencial sobre phishing: Como funciona e como se proteger**. 2021. Disponível em: https://www.avast.com/pt-br/c-phishing. Acesso em: 25 set. 2021.

Centenaro, M. *et al.* Long-range communications in unlicensed bands: the rising stars in the iot and smart city scenarios. **IEEE Wireless Communications**, v. 23, n. 5, p. 60–67, October 2016.

CHAUDHARI, B. S.; ZENNARO, M.; BORKAR, S. **LPWAN technologies: Emerging application characteristics, requirements, and design considerations**. [*S.l.*: *s.n.*], 2020. v. 12. ISSN 19995903.

CISCO. **Cisco Annual Internet Report (2018-2023) White Paper**. 2021. Disponível em: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. Acesso em: 27 jun. 2021.

CISCO. **O que é um firewall?** 2021. Disponível em: https://www.cisco.com/c/pt_br/products/security/firewalls/what-is-a-firewall.html. Acesso em: 15 set. 2021.

COMER, D. E. **Redes de computadores e Internet**: 6ªedição. São Paulo: Bookman Editora, 2016.

Connectivity Standards Alliance. **What is Zigbee?** 2021. Disponível em: https://zigbeealliance.org/solution/zigbee/. Acesso em: 20 set. 2021.

CPLUSPLUS. **Function Clock Reference**. 2021. Disponível em: https://www.cplusplus.com/reference/ctime/clock/. Acesso em: 13 out. 2021.

Cytron Technologies. **915MHz 8-channel LoRa Gateway Hat for Raspberry Pi**. 2021. Disponível em: https://www.robotshop.com/media/files/pdf/915mhz-lora-gateway-raspberry-pi-hat-datasheet2.pdf. Acesso em: 16 set. 2021.

Fortinet. **Latin america suffered more than 41 billion cyberattack attemps in 2020**. 2021. Disponível em: https://www.fortinet.com/br/corporate/about-us/newsroom/press-releases/2021/

latin-america-suffered-more-than-41-billion-cyberattack-attempts-in-2020. Acesso em: 26 jun. 2021.

FOUNDATION, P. S. **About Python**. 2021. Disponível em: https://www.python.org/about/. Acesso em: 20 out. 2021.

GNU. **The GNU C Reference Manual**. 2021. Disponível em: https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html. Acesso em: 20 out. 2021.

GUBBI, J. *et al.* Internet of things (iot): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, v. 29, n. 7, p. 1645 – 1660, 2013. ISSN 0167-739X. Disponível em: http://www.sciencedirect.com/science/article/pii/S0167739X13000241.

GUERRA, M. S. Ferramenta para análise de regras de firewall. **Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação)**, 2019.

HAT, R. **IPTables**. 2021. Disponível em: https://access.redhat.com/documentation/pt-br/red_hat_enterprise_linux/6/html/security_guide/sect-security_guide-iptables. Acesso em: 13 out. 2021.

Hoperf. **Low Power Long Range Transceiver Module**. 2021. Disponível em: https://cdn.sparkfun.com/assets/learn_tutorials/8/0/4/RFM95_96_97_98W.pdf. Acesso em: 20 set. 2021.

IBM. **O que é a tecnologia blockchain?** 2021. Disponível em: https://www.ibm.com/br-pt/topics/what-is-blockchain. Acesso em: 19 out. 2021.

IT MIDIA. **LoRa garante a conectividade da Agricultura de Precisão**. 2021. Disponível em: https://cio.com.br/gestao/lora-garante-a-conectividade-da-agricultura-de-precisao/. Acesso em: 20 out. 2021.

KASPERSKY. **O que é cibersegurança?** 2021. Disponível em: https://www.kaspersky.com.br/resource-center/definitions/what-is-cyber-security. Acesso em: 15 set. 2021.

KASPERSKY. **O que é um firewall?** 2021. Disponível em: https://www.kaspersky.com.br/resource-center/definitions/firewall. Acesso em: 15 set. 2021.

Krzysztof Gabis. **Parson**. 2021. Disponível em: https://github.com/kgabis/parson. Acesso em: 17 out. 2021.

Lavric, A.; Petrariu, A. I. Lorawan communication protocol: The new era of iot. *In*: **2018 International Conference on Development and Application Systems (DAS)**. [*S.l.*: *s.n.*], 2018. p. 74–77.

LORA ALLIANCE. **About LoRaWAN**. 2021. Disponível em: https://lora-alliance.org/about-lorawan/. Acesso em: 21 set. 2021.

LORA ALLIANCE. **LoRaWAN 1.0.3 Specification**. 2021. Disponível em: https://lora-alliance.org/sites/default/files/2018-07/lorawan1.0.3.pdf. Acesso em: 21 set. 2021.

LORA ALLIANCE. **LoRaWAN Coverage**. 2021. Disponível em: https://lora-alliance.org. Acesso em: 21 set. 2021.

LORA ALLIANCE. **LoRaWAN Especificação L2 1.0.4**. 2021. Disponível em: https://lora-alliance.org/resource_hub/lorawan-104-specification-package/. Acesso em: 23 set. 2021.

MALAN, J. *et al.* Framing the nature and scale of cyber security vulnerabilities within the current consumer internet of things (iot) landscape. **Centre for Strategy Evaluation Services**, 2020.

Meddeb, A. Internet of things standards: who stands out from the crowd? **IEEE Communications Magazine**, v. 54, n. 7, p. 40–47, July 2016.

MORAES, A. F. de. **Firewalls: segurança no controle de acesso**. São Paulo: Pearson, 2015.

NETFILTER. **The netfilter.org project**. 2021. Disponível em: https://www.netfilter.org/. Acesso em: 26 set. 2021.

Palattella, M. R. *et al.* Internet of things in the 5g era: Enablers, architecture, and business models. **IEEE Journal on Selected Areas in Communications**, v. 34, n. 3, p. 510–527, March 2016.

RASH, M. **Linux Firewalls**: Attack detection and response with iptables, psad, and fwsnort. San Francisco: William Pollock, 2007.

Raspberry. **Raspberry Pi Documentation**. 2021. Disponível em: https://www.raspberrypi.org/documentation/. Acesso em: 20 set. 2021.

SEMTECH. LoRa™Modulation Basics Semtech. AN1200.22, p. 1–26, 2015.

SEMTECH. **What are LoRa® and LoRaWAN®?** 2021. Disponível em: https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan. Acesso em: 26 set. 2021.

SIGFOX. **What is SIGFOX?** 2021. Disponível em: https://www.sigfox.com/en. Acesso em: 20 set. 2021.

TANENBAUM, A. S.; WETHERALL, D. **Redes de computadores**: 5ªedição. São Paulo: Pearson, 2011.

TECHNOLOGIES, C. **About Us**. 2021. Disponível em: https://www.cytron.io/about-us. Acesso em: 28 set. 2021.

TECHNOLOGIES, C. **LoRa Gateway**. 2021. Disponível em: https://github.com/CytronTechnologies/lora_gateway. Acesso em: 29 set. 2021.

TECHNOLOGIES, C. **Packet Forward**. 2021. Disponível em: https://github.com/CytronTechnologies/packet_forwarder. Acesso em: 29 set. 2021.

TECHNOLOGIES, C. **RisingHF-gateway**. 2021. Disponível em: https://github.com/CytronTechnologies/RisingHF-gateway. Acesso em: 29 set. 2021.

The Things Network. **Connect devices to The Things Network**. 2021. Disponível em: https://www.thethingsnetwork.org/docs/devices/. Acesso em: 20 set. 2021.

Yang, X. *et al.* Security vulnerabilities in lorawan. *In*: **2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)**. [*S.l.*: *s.n.*], 2018. p. 129–140.